libopusenc

0.3

Generated by Doxygen 1.14.0

Chapter 1

libopusenc

1.1 Introduction

This is the documentation for the `libopusenc` C API.

The `libopusenc` package provides a convenient high-level API for encoding Ogg Opus files.

1.2 Organization

The main API is divided into several sections:

- Encoding
- Comments Handling
- Encoding Options
- Callback Functions
- Error Codes

1.3 Overview

The `libopusfile` API provides an easy way to encode Ogg Opus files using `libopus`.

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Topic Documentation

5.1 Error Codes

List of possible error codes

Many of the functions in this library return a negative error code when a function fails.

This list provides a brief explanation of the common errors. See each individual function for more details on what a specific error code means in that context.

- #define OPE_API_VERSION 0
  API version for this header.
- #define **OPE_OK** 0
- #define **OPE_BAD_ARG** -11
- #define **OPE_INTERNAL_ERROR** -13
- #define **OPE_UNIMPLEMENTED** -15
- #define **OPE_ALLOC_FAIL** -17
- #define **OPE_CANNOT_OPEN** -30
- #define **OPE_TOO_LATE** -31
- #define **OPE_INVALID_PICTURE** -32
- #define **OPE_INVALID_ICON** -33
- #define **OPE_WRITE_FAIL** -34
- #define **OPE_CLOSE_FAIL** -35

5.1.1 Detailed Description

5.1.2 Macro Definition Documentation

5.1.2.1 OPE_API_VERSION

```
#define OPE_API_VERSION 0
```

API version for this header.

Can be used to check for features at compile time.

5.2 Encoding Options

Control parameters

Macros for setting encoder options.

- #define OPE_SET_DECISION_DELAY(x)
- #define OPE_GET_DECISION_DELAY(x)
- #define OPE_SET_MUXING_DELAY(x)
- #define OPE_GET_MUXING_DELAY(x)
- #define OPE_SET_COMMENT_PADDING(x)
- #define OPE_GET_COMMENT_PADDING(x)
- #define OPE_SET_SERIALNO(x)
- #define OPE_GET_SERIALNO(x)
- #define OPE_SET_PACKET_CALLBACK(x, u)
- #define OPE_SET_HEADER_GAIN(x)
- #define OPE_GET_HEADER_GAIN(x)
- #define OPE_GET_NB_STREAMS(x)
- #define OPE_GET_NB_COUPLED_STREAMS(x)

5.2.1 Detailed Description

5.2.2 Macro Definition Documentation

5.2.2.1 OPE_SET_DECISION_DELAY

```
#define OPE_SET_DECISION_DELAY(
              x)
```

Value:

```
OPE_SET_DECISION_DELAY_REQUEST, ope_check_int(x)
```

5.2.2.2 OPE_GET_DECISION_DELAY

```
#define OPE_GET_DECISION_DELAY(
              x)
```

Value:

```
OPE_GET_DECISION_DELAY_REQUEST, ope_check_int_ptr(x)
```

5.2.2.3 OPE_SET_MUXING_DELAY

```
#define OPE_SET_MUXING_DELAY(
              x)
```

Value:

```
OPE_SET_MUXING_DELAY_REQUEST, ope_check_int(x)
```

5.2.2.4 OPE_GET_MUXING_DELAY

```
#define OPE_GET_MUXING_DELAY(
              x)
```

Value:

```
OPE_GET_MUXING_DELAY_REQUEST, ope_check_int_ptr(x)
```

5.2.2.5 OPE_SET_COMMENT_PADDING

```
#define OPE_SET_COMMENT_PADDING(
              x)
```

Value:

```
OPE_SET_COMMENT_PADDING_REQUEST, ope_check_int(x)
```

5.2.2.6 OPE_GET_COMMENT_PADDING

```
#define OPE_GET_COMMENT_PADDING(
              x)
```

Value:

```
OPE_GET_COMMENT_PADDING_REQUEST, ope_check_int_ptr(x)
```

5.2.2.7 OPE_SET_SERIALNO

```
#define OPE_SET_SERIALNO(
              x)
```

Value:

```
OPE_SET_SERIALNO_REQUEST, ope_check_int(x)
```

5.2.2.8 OPE_GET_SERIALNO

```
#define OPE_GET_SERIALNO(
              x)
```

Value:

```
OPE_GET_SERIALNO_REQUEST, ope_check_int_ptr(x)
```

5.2.2.9 OPE_SET_PACKET_CALLBACK

```
#define OPE_SET_PACKET_CALLBACK(
              x,
              u)
```

Value:

```
OPE_SET_PACKET_CALLBACK_REQUEST, ope_check_packet_func(x), ope_check_void_ptr(u)
```

5.2.2.10 OPE_SET_HEADER_GAIN

```
#define OPE_SET_HEADER_GAIN(
              x)
```

Value:

```
OPE_SET_HEADER_GAIN_REQUEST, ope_check_int(x)
```

5.2.2.11 OPE_GET_HEADER_GAIN

```
#define OPE_GET_HEADER_GAIN(
              x)
```

Value:

```
OPE_GET_HEADER_GAIN_REQUEST, ope_check_int_ptr(x)
```

5.2.2.12 OPE_GET_NB_STREAMS

```
#define OPE_GET_NB_STREAMS(
              x)
```

Value:

```
OPE_GET_NB_STREAMS_REQUEST, ope_check_int_ptr(x)
```

5.2.2.13 OPE_GET_NB_COUPLED_STREAMS

```
#define OPE_GET_NB_COUPLED_STREAMS(
              x)
```

Value:

```
OPE_GET_NB_COUPLED_STREAMS_REQUEST, ope_check_int_ptr(x)
```

5.3 Callback Functions

Data Structures

- struct OpusEncCallbacks

  Callback functions for accessing the stream.

Callback functions

These are the callbacks that can be implemented for an encoder.

- typedef int(∗ ope_write_func) (void ∗user_data, const unsigned char ∗ptr, opus_int32 len)

  Called for writing a page.
- typedef int(∗ ope_close_func) (void ∗user_data)

  Called for closing a stream.
- typedef void(∗ ope_packet_func) (void ∗user_data, const unsigned char ∗packet_ptr, opus_int32 packet_len, opus_uint32 flags)

  Called on every packet encoded (including header).

5.3.1 Detailed Description

5.3.2 Typedef Documentation

5.3.2.1 ope_write_func

```
typedef int(* ope_write_func) (void *user_data, const unsigned char *ptr, opus_int32 len)
```

Called for writing a page.

Parameters

user_data	user-defined data passed to the callback
ptr	buffer to be written
len	number of bytes to be written

Returns

error code

Return values

0	success
1	failure

5.3.2.2 ope_close_func

```
typedef int(* ope_close_func) (void *user_data)
```

Called for closing a stream.

Parameters

user_data	user-defined data passed to the callback

Returns

error code

Return values

0	success
1	failure

5.3.2.3 ope_packet_func

```
typedef void(* ope_packet_func) (void *user_data, const unsigned char *packet_ptr, opus_int32
packet_len, opus_uint32 flags)
```

Called on every packet encoded (including header).

Parameters

user_data	user-defined data passed to the callback
packet_ptr	packet data
packet_len	number of bytes in the packet
flags	optional flags (none defined for now so zero)

5.4 Comments Handling

Functions for handling comments

These functions make it possible to add comments and pictures to Ogg Opus files.

- OPE_EXPORT OggOpusComments ∗ ope_comments_create (void)
  Create a new comments object.
- OPE_EXPORT OggOpusComments ∗ ope_comments_copy (OggOpusComments ∗comments)
  Create a deep copy of a comments object.
- OPE_EXPORT void ope_comments_destroy (OggOpusComments ∗comments)
  Destroys a comments object.
- OPE_EXPORT int ope_comments_add (OggOpusComments ∗comments, const char ∗tag, const char ∗val)
  Add a comment.
- OPE_EXPORT int ope_comments_add_string (OggOpusComments ∗comments, const char ∗tag_and_val)
  Add a comment as a single tag=value string.
- OPE_EXPORT int ope_comments_add_picture (OggOpusComments ∗comments, const char ∗filename, int picture_type, const char ∗description)
  Add a picture from a file.
- OPE_EXPORT int ope_comments_add_picture_from_memory (OggOpusComments ∗comments, const char ∗ptr, size_t size, int picture_type, const char ∗description)
  Add a picture already in memory.

5.4.1 Detailed Description

5.4.2 Function Documentation

5.4.2.1 ope_comments_create()

```
OPE_EXPORT OggOpusComments * ope_comments_create (
            void )
```

Create a new comments object.

Returns

Newly-created comments object.

5.4.2.2 ope_comments_copy()

```
OPE_EXPORT OggOpusComments * ope_comments_copy (
            OggOpusComments * comments)
```

Create a deep copy of a comments object.

Parameters

comments	Comments object to copy

Returns

Deep copy of input.

5.4.2.3 ope_comments_destroy()

```
OPE_EXPORT void ope_comments_destroy (
            OggOpusComments * comments)
```

Destroys a comments object.

Parameters

comments	Comments object to destroy

5.4.2.4 ope_comments_add()

```
OPE_EXPORT int ope_comments_add (
            OggOpusComments * comments,
            const char * tag,
            const char * val)
```

Add a comment.

Parameters

in,out	*comments*	Where to add the comments
	tag	Tag for the comment (must not contain = char)
	val	Value for the tag

Returns

Error code

5.4.2.5 ope_comments_add_string()

```
OPE_EXPORT int ope_comments_add_string (
            OggOpusComments * comments,
            const char * tag_and_val)
```

Add a comment as a single tag=value string.

Parameters

in,out	*comments*	Where to add the comments
	tag_and_val	string of the form tag=value (must contain = char)

Returns

Error code

5.4.2.6 ope_comments_add_picture()

```
OPE_EXPORT int ope_comments_add_picture (
            OggOpusComments * comments,
            const char * filename,
            int picture_type,
            const char * description)
```

Add a picture from a file.

Parameters

in,out	*comments*	Where to add the comments
	filename	File name for the picture
	picture_type	Type of picture (-1 for default)
	description	Description (NULL means no comment)

Returns

Error code

5.4.2.7 ope_comments_add_picture_from_memory()

```
OPE_EXPORT int ope_comments_add_picture_from_memory (
            OggOpusComments * comments,
            const char * ptr,
            size_t size,
            int picture_type,
            const char * description)
```

Add a picture already in memory.

Parameters

in,out	*comments*	Where to add the comments
	ptr	Pointer to picture in memory
	size	Size of picture pointed to by ptr
	picture_type	Type of picture (-1 for default)
	description	Description (NULL means no comment)

Returns

Error code

5.5 Encoding

Functions for encoding Ogg Opus files

These functions make it possible to encode Ogg Opus files.

- OPE_EXPORT OggOpusEnc * ope_encoder_create_file (const char *path, OggOpusComments *comments, opus_int32 rate, int channels, int family, int *error)

  Create a new OggOpus file.
- OPE_EXPORT OggOpusEnc * ope_encoder_create_callbacks (const OpusEncCallbacks *callbacks, void *user_data, OggOpusComments *comments, opus_int32 rate, int channels, int family, int *error)

  Create a new OggOpus stream to be handled using callbacks.
- OPE_EXPORT OggOpusEnc * ope_encoder_create_pull (OggOpusComments *comments, opus_int32 rate, int channels, int family, int *error)

  Create a new OggOpus stream to be used along with.ope_encoder_get_page().
- OPE_EXPORT int ope_encoder_deferred_init_with_mapping (OggOpusEnc *enc, int family, int streams, int coupled_streams, const unsigned char *mapping)

  Deferred initialization of the encoder to force an explicit channel mapping.
- OPE_EXPORT int ope_encoder_write_float (OggOpusEnc *enc, const float *pcm, int samples_per_channel)

  Add/encode any number of float samples to the stream.
- OPE_EXPORT int ope_encoder_write (OggOpusEnc *enc, const opus_int16 *pcm, int samples_per_↩ channel)

  Add/encode any number of 16-bit linear samples to the stream.
- OPE_EXPORT int ope_encoder_get_page (OggOpusEnc *enc, unsigned char **page, opus_int32 *len, int flush)

  Get the next page from the stream (only if using ope_encoder_create_pull()).
- OPE_EXPORT int ope_encoder_drain (OggOpusEnc *enc)

  Finalizes the stream, but does not deallocate the object.
- OPE_EXPORT void ope_encoder_destroy (OggOpusEnc *enc)

  Deallocates the obect.
- OPE_EXPORT int ope_encoder_chain_current (OggOpusEnc *enc, OggOpusComments *comments)

  Ends the stream and create a new stream within the same file.
- OPE_EXPORT int ope_encoder_continue_new_file (OggOpusEnc *enc, const char *path, OggOpus↩ Comments *comments)

  Ends the stream and create a new file.
- OPE_EXPORT int ope_encoder_continue_new_callbacks (OggOpusEnc *enc, void *user_data, OggOpus↩ Comments *comments)

  Ends the stream and create a new file (callback-based).
- OPE_EXPORT int ope_encoder_flush_header (OggOpusEnc *enc)

  Write out the header now rather than wait for audio to begin.
- OPE_EXPORT int ope_encoder_ctl (OggOpusEnc *enc, int request,...)

  Sets encoder options.
- OPE_EXPORT const char * ope_strerror (int error)

  Converts a libopusenc error code into a human readable string.
- OPE_EXPORT const char * ope_get_version_string (void)

  Returns a string representing the version of libopusenc being used at run time.
- OPE_EXPORT int ope_get_abi_version (void)

  ABI version for this header.

5.5.1 Detailed Description

5.5.2 Function Documentation

5.5.2.1 ope_encoder_create_file()

```
OPE_EXPORT OggOpusEnc * ope_encoder_create_file (
            const char * path,
            OggOpusComments * comments,
            opus_int32 rate,
            int channels,
            int family,
            int * error)
```

Create a new OggOpus file.

Parameters

	path	Path where to create the file
	comments	Comments associated with the stream
	rate	Input sampling rate (48 kHz is faster)
	channels	Number of channels
	family	Mapping family (0 for mono/stereo, 1 for surround)
out	*error*	Error code (NULL if no error is to be returned)

Returns

Newly-created encoder.

5.5.2.2 ope_encoder_create_callbacks()

```
OPE_EXPORT OggOpusEnc * ope_encoder_create_callbacks (
            const OpusEncCallbacks * callbacks,
            void * user_data,
            OggOpusComments * comments,
            opus_int32 rate,
            int channels,
            int family,
            int * error)
```

Create a new OggOpus stream to be handled using callbacks.

Parameters

	callbacks	Callback functions
	user_data	Pointer to be associated with the stream and passed to the callbacks
	comments	Comments associated with the stream
	rate	Input sampling rate (48 kHz is faster)
	channels	Number of channels
	family	Mapping family (0 for mono/stereo, 1 for surround)
out	*error*	Error code (NULL if no error is to be returned)

Returns

Newly-created encoder.

5.5.2.3 ope_encoder_create_pull()

```
OPE_EXPORT OggOpusEnc * ope_encoder_create_pull (
             OggOpusComments * comments,
             opus_int32 rate,
             int channels,
             int family,
             int * error)
```

Create a new OggOpus stream to be used along with.ope_encoder_get_page().

This is mostly useful for muxing with other streams.

Parameters

	comments	Comments associated with the stream
	rate	Input sampling rate (48 kHz is faster)
	channels	Number of channels
	family	Mapping family (0 for mono/stereo, 1 for surround)
out	*error*	Error code (NULL if no error is to be returned)

Returns

Newly-created encoder.

5.5.2.4 ope_encoder_deferred_init_with_mapping()

```
OPE_EXPORT int ope_encoder_deferred_init_with_mapping (
             OggOpusEnc * enc,
             int family,
             int streams,
             int coupled_streams,
             const unsigned char * mapping)
```

Deferred initialization of the encoder to force an explicit channel mapping.

This can be used to override the default channel coupling, but using it for regular surround will almost certainly lead to worse quality.

Parameters

in,out	*enc*	Encoder
	family	Mapping family (0 for mono/stereo, 1 for surround)
	streams	Total number of streams
	coupled_streams	Number of coupled streams
	mapping	Channel mapping

Returns

Error code

5.5.2.5 ope_encoder_write_float()

```
OPE_EXPORT int ope_encoder_write_float (
            OggOpusEnc * enc,
            const float * pcm,
            int samples_per_channel)
```

Add/encode any number of float samples to the stream.

Parameters

in,out	*enc*	Encoder
	pcm	Floating-point PCM values in the +/-1 range (interleaved if multiple channels)
	samples_per_channel	Number of samples for each channel

Returns

Error code

5.5.2.6 ope_encoder_write()

```
OPE_EXPORT int ope_encoder_write (
            OggOpusEnc * enc,
            const opus_int16 * pcm,
            int samples_per_channel)
```

Add/encode any number of 16-bit linear samples to the stream.

Parameters

in,out	*enc*	Encoder
	pcm	Linear 16-bit PCM values in the [-32768,32767] range (interleaved if multiple channels)
	samples_per_channel	Number of samples for each channel

Returns

Error code

5.5.2.7 ope_encoder_get_page()

```
OPE_EXPORT int ope_encoder_get_page (
            OggOpusEnc * enc,
            unsigned char ** page,
            opus_int32 * len,
            int flush)
```

Get the next page from the stream (only if using ope_encoder_create_pull()).

Parameters

in,out	*enc*	Encoder
out	*page*	Next available encoded page
out	*len*	Size (in bytes) of the page returned
	flush	If non-zero, forces a flush of the page (if any data avaiable)

Returns

1 if there is a page available, 0 if not.

5.5.2.8 ope_encoder_drain()

```
OPE_EXPORT int ope_encoder_drain (
            OggOpusEnc * enc)
```

Finalizes the stream, but does not deallocate the object.

Parameters

in,out	*enc*	Encoder

Returns

Error code

5.5.2.9 ope_encoder_destroy()

```
OPE_EXPORT void ope_encoder_destroy (
            OggOpusEnc * enc)
```

Deallocates the obect.

Make sure to ope_drain() first.

Parameters

in,out	*enc*	Encoder

5.5.2.10 ope_encoder_chain_current()

```
OPE_EXPORT int ope_encoder_chain_current (
            OggOpusEnc * enc,
            OggOpusComments * comments)
```

Ends the stream and create a new stream within the same file.

Parameters

in,out	*enc*	Encoder
	comments	Comments associated with the stream

Returns

Error code

5.5.2.11 ope_encoder_continue_new_file()

```
OPE_EXPORT int ope_encoder_continue_new_file (
            OggOpusEnc * enc,
            const char * path,
            OggOpusComments * comments)
```

Ends the stream and create a new file.

Parameters

in,out	*enc*	Encoder
	path	Path where to write the new file
	comments	Comments associated with the stream

Returns

Error code

5.5.2.12 ope_encoder_continue_new_callbacks()

```
OPE_EXPORT int ope_encoder_continue_new_callbacks (
            OggOpusEnc * enc,
            void * user_data,
            OggOpusComments * comments)
```

Ends the stream and create a new file (callback-based).

Parameters

in,out	*enc*	Encoder
	user_data	Pointer to be associated with the new stream and passed to the callbacks
	comments	Comments associated with the stream

Returns

Error code

5.5.2.13 ope_encoder_flush_header()

```
OPE_EXPORT int ope_encoder_flush_header (
            OggOpusEnc * enc)
```

Write out the header now rather than wait for audio to begin.

Parameters

in,out	*enc*	Encoder

Returns

Error code

5.5.2.14 ope_encoder_ctl()

```
OPE_EXPORT int ope_encoder_ctl (
            OggOpusEnc * enc,
            int request,
             ...)
```

Sets encoder options.

Parameters

in,out	*enc*	Encoder
	request	Use a request macro

Returns

Error code

5.5.2.15 ope_strerror()

```
OPE_EXPORT const char * ope_strerror (
            int error)
```

Converts a libopusenc error code into a human readable string.

Parameters

error	Error number

Returns

Error string

5.5.2.16 ope_get_version_string()

```
OPE_EXPORT const char * ope_get_version_string (
            void )
```

Returns a string representing the version of libopusenc being used at run time.

Returns

A string describing the version of this library

5.5.2.17 ope_get_abi_version()

```
OPE_EXPORT int ope_get_abi_version (
            void )
```

ABI version for this header.

Can be used to check for features at run time.

Returns

An integer representing the ABI version

Chapter 6

Data Structure Documentation

6.1 OpusEncCallbacks Struct Reference

Callback functions for accessing the stream.

`#include <opusenc.h>`

Data Fields

- ope_write_func **write**

  Callback for writing to the stream.
- ope_close_func **close**

  Callback for closing the stream.

6.1.1 Detailed Description

Callback functions for accessing the stream.

The documentation for this struct was generated from the following file:

- opusenc.h

Chapter 7

File Documentation

7.1 opusenc.h

```
00001 /* Copyright (c) 2017 Jean-Marc Valin */
00002 /*
00003    Redistribution and use in source and binary forms, with or without
00004    modification, are permitted provided that the following conditions
00005    are met:
00006
00007    - Redistributions of source code must retain the above copyright
00008    notice, this list of conditions and the following disclaimer.
00009
00010    - Redistributions in binary form must reproduce the above copyright
00011    notice, this list of conditions and the following disclaimer in the
00012    documentation and/or other materials provided with the distribution.
00013
00014    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00015    ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00016    LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00017    A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
00018    OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00019    EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00020    PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00021    PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
00022    LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00023    NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00024    SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00025 */
00026
00027 #ifndef OPUSENC_H
00028 # define OPUSENC_H
00029
00052
00053 # if defined(__cplusplus)
00054 extern "C" {
00055 # endif
00056
00057 #include <stddef.h>
00058 #include <opus.h>
00059
00060 #ifndef OPE_EXPORT
00061 # if defined(WIN32)
00062 #  if defined(OPE_BUILD) && defined(DLL_EXPORT)
00063 #   define OPE_EXPORT __declspec(dllexport)
00064 #  else
00065 #   define OPE_EXPORT
00066 #  endif
00067 # elif defined(__GNUC__) && defined(OPE_BUILD)
00068 #  define OPE_EXPORT __attribute__ ((visibility ("default")))
00069 # else
00070 #  define OPE_EXPORT
00071 # endif
00072 #endif
00073
00083
00084
00085 /* Bump this when we change the API. */
00087 #define OPE_API_VERSION 0
00088
00089 #define OPE_OK 0
```

```
00090 /* Based on the relevant libopus code minus 10. */
00091 #define OPE_BAD_ARG -11
00092 #define OPE_INTERNAL_ERROR -13
00093 #define OPE_UNIMPLEMENTED -15
00094 #define OPE_ALLOC_FAIL -17
00095
00096 /* Specific to libopusenc. */
00097 #define OPE_CANNOT_OPEN -30
00098 #define OPE_TOO_LATE -31
00099 #define OPE_INVALID_PICTURE -32
00100 #define OPE_INVALID_ICON -33
00101 #define OPE_WRITE_FAIL -34
00102 #define OPE_CLOSE_FAIL -35
00103
00106
00107 /* These are the "raw" request values -- they should usually not be used. */
00108 #define OPE_SET_DECISION_DELAY_REQUEST      14000
00109 #define OPE_GET_DECISION_DELAY_REQUEST      14001
00110 #define OPE_SET_MUXING_DELAY_REQUEST        14002
00111 #define OPE_GET_MUXING_DELAY_REQUEST        14003
00112 #define OPE_SET_COMMENT_PADDING_REQUEST     14004
00113 #define OPE_GET_COMMENT_PADDING_REQUEST     14005
00114 #define OPE_SET_SERIALNO_REQUEST            14006
00115 #define OPE_GET_SERIALNO_REQUEST            14007
00116 #define OPE_SET_PACKET_CALLBACK_REQUEST     14008
00117 /*#define OPE_GET_PACKET_CALLBACK_REQUEST     14009*/
00118 #define OPE_SET_HEADER_GAIN_REQUEST         14010
00119 #define OPE_GET_HEADER_GAIN_REQUEST         14011
00120 #define OPE_GET_NB_STREAMS_REQUEST          14013
00121 #define OPE_GET_NB_COUPLED_STREAMS_REQUEST  14015
00122
00123 /* Macros to trigger compilation errors when the wrong types are provided to a CTL. */
00124 /* These macros are not part of the API and are only for use within the macros below. */
00125 #define ope_check_int(x) (((void)((x) == (opus_int32)0)), (opus_int32)(x))
00126 #define ope_check_int_ptr(ptr) ((ptr) + ((ptr) - (opus_int32*)(ptr)))
00127 #define ope_check_packet_func(x) ((void)((void (*)(void *, const unsigned char *, opus_int32,
      opus_uint32))0 == (x)), (x))
00128 #define ope_check_void_ptr(x) ((void)((void *)0 == (x)), (x))
00129
00132
00137
00138 #define OPE_SET_DECISION_DELAY(x) OPE_SET_DECISION_DELAY_REQUEST, ope_check_int(x)
00139 #define OPE_GET_DECISION_DELAY(x) OPE_GET_DECISION_DELAY_REQUEST, ope_check_int_ptr(x)
00140 #define OPE_SET_MUXING_DELAY(x) OPE_SET_MUXING_DELAY_REQUEST, ope_check_int(x)
00141 #define OPE_GET_MUXING_DELAY(x) OPE_GET_MUXING_DELAY_REQUEST, ope_check_int_ptr(x)
00142 #define OPE_SET_COMMENT_PADDING(x) OPE_SET_COMMENT_PADDING_REQUEST, ope_check_int(x)
00143 #define OPE_GET_COMMENT_PADDING(x) OPE_GET_COMMENT_PADDING_REQUEST, ope_check_int_ptr(x)
00144 #define OPE_SET_SERIALNO(x) OPE_SET_SERIALNO_REQUEST, ope_check_int(x)
00145 #define OPE_GET_SERIALNO(x) OPE_GET_SERIALNO_REQUEST, ope_check_int_ptr(x)
00146 #define OPE_SET_PACKET_CALLBACK(x,u) OPE_SET_PACKET_CALLBACK_REQUEST, ope_check_packet_func(x),
      ope_check_void_ptr(u)
00147 /*#define OPE_GET_PACKET_CALLBACK(x,u) OPE_GET_PACKET_CALLBACK_REQUEST, (x), (u)*/
00148 #define OPE_SET_HEADER_GAIN(x) OPE_SET_HEADER_GAIN_REQUEST, ope_check_int(x)
00149 #define OPE_GET_HEADER_GAIN(x) OPE_GET_HEADER_GAIN_REQUEST, ope_check_int_ptr(x)
00150 #define OPE_GET_NB_STREAMS(x) OPE_GET_NB_STREAMS_REQUEST, ope_check_int_ptr(x)
00151 #define OPE_GET_NB_COUPLED_STREAMS(x) OPE_GET_NB_COUPLED_STREAMS_REQUEST, ope_check_int_ptr(x)
00154
00157
00162
00171 typedef int (*ope_write_func)(void *user_data, const unsigned char *ptr, opus_int32 len);
00172
00179 typedef int (*ope_close_func)(void *user_data);
00180
00187 typedef void (*ope_packet_func)(void *user_data, const unsigned char *packet_ptr, opus_int32
      packet_len, opus_uint32 flags);
00188
00190 typedef struct {
00192   ope_write_func write;
00194   ope_close_func close;
00195 } OpusEncCallbacks;
00198
00200 typedef struct OggOpusComments OggOpusComments;
00201
00203 typedef struct OggOpusEnc OggOpusEnc;
00204
00207
00212
00215 OPE_EXPORT OggOpusComments *ope_comments_create(void);
00216
00220 OPE_EXPORT OggOpusComments *ope_comments_copy(OggOpusComments *comments);
00221
00224 OPE_EXPORT void ope_comments_destroy(OggOpusComments *comments);
00225
00232 OPE_EXPORT int ope_comments_add(OggOpusComments *comments, const char *tag, const char *val);
00233
00239 OPE_EXPORT int ope_comments_add_string(OggOpusComments *comments, const char *tag_and_val);
00240
```

```
00248 OPE_EXPORT int ope_comments_add_picture(OggOpusComments *comments, const char *filename, int
      picture_type, const char *description);
00249
00258 OPE_EXPORT int ope_comments_add_picture_from_memory(OggOpusComments *comments, const char *ptr, size_t
      size, int picture_type, const char *description);
00259
00262
00265
00270
00280 OPE_EXPORT OggOpusEnc *ope_encoder_create_file(const char *path, OggOpusComments *comments, opus_int32
      rate, int channels, int family, int *error);
00281
00292 OPE_EXPORT OggOpusEnc *ope_encoder_create_callbacks(const OpusEncCallbacks *callbacks, void
      *user_data,
00293     OggOpusComments *comments, opus_int32 rate, int channels, int family, int *error);
00294
00304 OPE_EXPORT OggOpusEnc *ope_encoder_create_pull(OggOpusComments *comments, opus_int32 rate, int
      channels, int family, int *error);
00305
00315 OPE_EXPORT int ope_encoder_deferred_init_with_mapping(OggOpusEnc *enc, int family, int streams,
00316     int coupled_streams, const unsigned char *mapping);
00317
00323 OPE_EXPORT int ope_encoder_write_float(OggOpusEnc *enc, const float *pcm, int samples_per_channel);
00324
00330 OPE_EXPORT int ope_encoder_write(OggOpusEnc *enc, const opus_int16 *pcm, int samples_per_channel);
00331
00338 OPE_EXPORT int ope_encoder_get_page(OggOpusEnc *enc, unsigned char **page, opus_int32 *len, int
      flush);
00339
00344 OPE_EXPORT int ope_encoder_drain(OggOpusEnc *enc);
00345
00349 OPE_EXPORT void ope_encoder_destroy(OggOpusEnc *enc);
00350
00356 OPE_EXPORT int ope_encoder_chain_current(OggOpusEnc *enc, OggOpusComments *comments);
00357
00364 OPE_EXPORT int ope_encoder_continue_new_file(OggOpusEnc *enc, const char *path, OggOpusComments
      *comments);
00365
00372 OPE_EXPORT int ope_encoder_continue_new_callbacks(OggOpusEnc *enc, void *user_data, OggOpusComments
      *comments);
00373
00378 OPE_EXPORT int ope_encoder_flush_header(OggOpusEnc *enc);
00379
00385 OPE_EXPORT int ope_encoder_ctl(OggOpusEnc *enc, int request, ...);
00386
00392 OPE_EXPORT const char *ope_strerror(int error);
00393
00396 OPE_EXPORT const char *ope_get_version_string(void);
00397
00400 OPE_EXPORT int ope_get_abi_version(void);
00401
00404
00405 # if defined(__cplusplus)
00406 }
00407 # endif
00408
00409 #endif
```

Index